2011 Third International Conference on Intelligent Networking and Collaborative Systems

The evaluation of Transfer Time, CPU Consumption and Memory Utilization in XEN-PV, XEN-HVM, OpenVZ, KVM-FV and KVM-PV Hypervisors using FTP and HTTP approaches

Igli TAFA, Ermal BEQIRI, Hakik Paci, Elinda KAJO, Aleksandër XHUVANI.

Polytechnic University of Tirana, Information Technology Faculty, Tiranë, Albania

itafaj@fti.edu.al, ermalfr@yahoo.fr, hakikpaci@gmail.com , e_kajo@yahoo.com, axhuvani@yahoo.com

Abstract - In this paper the authors evaluate the CPU Consumption, Memory Utilization and Transfer Time between 5 Hypervisors XEN-PV, XEN-HVM, OpenVZ, KVM-FV and KVM-PV by using FTP and HTTP approaches. All the results are compared with the Real Environment. From the experimental results, the authors have concluded that OpenVZ and XEN-PV have better performance than other hypervisors. The worse performance is for KVM Hypervisor. In our paper we have used some scripts in order to evaluate the performance of these Hypervisors.

Keyword: XEN-PV, XEN-HVM, OpenVZ, KVM-FV, KVM-PV, FTP approach, HTTP approach.

1. INTRODUCTION

The virtualization is one of the hottest topics in nowadays distributed computation as they are one of the key elements in cloud computing. The aim is to improve the efficiency of resource utilization, to reduce the energy consumption, to increase the availability of applications etc. One of the most efficient techniques of virtualization is Live Migration. Based on this technique, one application can migrate from one machine to another with a minimal transfer time (about some ms) [1], [2] . In order to reach these facilities, it is offered a software layer called Hypervisor. There are some types of Hypervisors such as: Xen, OpenVZ, KVM, VMWare, VirtualBox etc. Each hypervisor has it`s significant characteristics and they are divided based on three significant virtualization techniques: Full Virtualization approach, Para Virtualization approach and Operating System Virtualization approach [3]. The advantages and disadvantages for the hypervisors are presented in table 1.

Table 1. The advantages and disadvantages of each Hypervisor

Full Virtualization	Para Virtualization	OS Virtualization
VmWare, Virtual Box, KVM, Xen-HVM	Xen, ESX-Server , KVM-PV	Open VZ
Hybrid GuestOS can be built over the Hypervisor	Only Linux GuestOS can be built over the hypervisor	Only Linux GuestOS can be built over the hypervisor
Hybrid Computer Architectures can communicate with each-other	Similar Computer Architectures can communicate with each other	Similar Computer Architectures can communicate with each other
Generally a Low Efficiency in I/O	Generally a High Efficiency in I/O	Generally a High Efficiency in I/O
Generally a High overhead	Generally a High overhead	Generally a Low overhead
Generally Big time delays during processing	Generally no time delays during processing	Generally Low time delays during processing

However these characteristics are not absolute, since in some cases there are some deviances. In [3] VMWare has a better performance than other hypervisors.

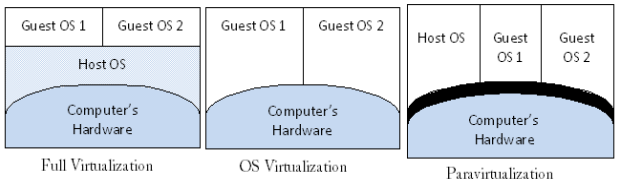

Fig1. Three types of Virtualization.

In this paper the authors evaluate the Transfer Time, CPU consumption and Memory Utilization by using FTP and HTTP protocols from a client machine to a server machine inside a LAN with 5 hypervisors: Xen-PV, Xen-HVM, OpenVZ, KVM-FV and KVM-PV. Finally it gives the comparison of the results with the case of a real machine.

The paper is organized as follows: In section 2 are presented the Related Works, in section 3 is introduced the Background, section 4 presents the Experimental Phase and in section 5 are given Conclusions & Future Works.

2. RELATED WORKS

In [4] is done a comparison between HPC in KVM and in OpenVZ. In this article is evaluated the throughput in reading and writing a SAS (Serial Attached SCSI) disk and is concluded that KVM has got a better performance in reading than OpenVZ, but it doesn't happen the same in writing to a disk. It is also tested the Round Trip-Time from one host to another with the same application above the hypervisors KVM and OpenVZ. Also an interesting comparison is the measurement of the Maximal Transmission Speed from a host to another referring to KVM-PV and KVM-FV. From the results is seen that KVM-PV has a better Transmission speed than KVM-FV.

978-0-7695-4579-0/11 $26.00 © 2011 IEEE
DOI 10.1109/INCoS.2011.164

In [5] is shown the difference between KVM, OpenVZ and XEN and is analyzed their performance using some benchmarks. Experiments present that KVM has lower CPU Processing than two remain hypervisors. This is because KVM has got a bigger overhead than other hypervisors. Big overhead is introduced from the complexity of QEMU emulator. Also we see that the process of copying data in memory is slower in KVM than in two other hypervisors. This happens because of API interface. Since KVM has got Full-Virtualization, the access and communication of GuestOS with the applications above it (which share the same interface) is slower. In [5] is measured the performance in reading and writing 3 SAS discs. From the experiments in [5] it can be seen that OpenVZ has got the highest speed, followed by Xen and finally by KVM Hypervisor. OpenVZ has got a good performance because the hypervisor introduces a smaller complexity then two other cases. GuestOS's in OpenVZ seem like processes which are executed in CPU and share the same kernel with the host.

In [3] are used different measurements using different tools . It is a module incorporated in Linux's kernel which serves to generate traffic from one host to another and evaluates the CPU consumption and memory utilization. It is noticed that OpenVZ has the best performance and KVM has the worst. In [3] the tests are performed sending different streams with different packet size from one virtual machine to another in a computer network connected with a gigabit switch, for three hypervisors,
In all the experiments, OpenVZ has the best performance followed by XEN.
Based on [6], Xen has a better scalability than KVM. KVM crashes for 3VM, 6VM, 10VM and 11VM above it. But anyway Xen has lower scalability than OpenVZ, because for OpenVZ the GuestOSs are processes, although they consume a lot of memory and processing [7].

In [8] is made a comparison of CPU performance for the same applications between XEN-PV, Xen-FV, KVM-FV and KVM-PV. It is seen that XEN-PV consumes less CPU. XEN-PV has the highest speed of writing SAS discs, and KVM-PV wich uses ''virtio'' drivers [8] has got the lowest. In terms of TCP throughput from a VM to a remote host, XEN-PV has the best performance followed by KVM-PV. This performance is increased because of KVM-PV has improved the speed of the communication interface between GuestOS and Kernel OS in ABI interface. KVM-FV consumes less bandwidth, and XEN-PV consumes more than others. KVM-FV has the biggest latency in copying the files in memory.
In all the papers that the authors have studied, there wasn't a evaluation of CPU consumption, Memory Utilization and Transfer Time using FTP and HTTP protocols. Thus the aim of this paper is to realize some experiments using some tools in order to evaluate these parameters in both protocols.

3. EXPERIMENTAL SETUP

Based on figure 1 we want to evaluate three parameters which are: Total Transfer Time, CPU consumption and Memory Utilization by using FTP and HTTP protocols between one virtual machine built in a host to another virtual machine located in another. The two virtual machines use the same Hypervisor. One machine serves as a Client Machine and the other as a Server Machine. Then we want to compare all these results with the case of real physical hosts (no hypervisor). As it looks from figure 2 we have used 5 kinds of hypervisors: XEN-PV, XEN-HVM, OpenVZ, KVM-FV, KVM-PV.

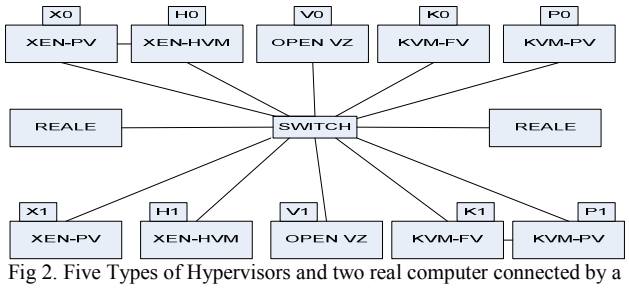

Fig 2. Five Types of Hypervisors and two real computer connected by a Gigabit Switch.

There are 12 computers connected by a Cisco Gigabit Switch with 16 gates. All the computers are connected by Fiber Channel communication and Fast Ethernet Interface 100/1000 Mbit/sec. The network topology is bus and all the packets are transmitted in the Layer 2 of OSI Architecture. All the computers are configured with C class IP, based on TCP protocol. Each client can communicate only with the server (no with other computers), for example x0 in Xen-PV is a client machine and x1 is the Server Machine. Both computers are not interested about other computers connected in LAN. The same situation will be for each pair of Client-Server machines. Host Operating System is CentOS 5.5 and GuestOS is Ubuntu 10.04 Server. All computers have the same architecture: Intel Core i7 920, Quad Core +, L2 4x256 KB, L3 = 8 MB, Asus, Three Channel DDR3 1600 Mhz, RAM 3x2GB, 64 bit processor, Hyperthread Technology, Freq 3.2 Ghz, VT Support, Turbo Boost Support.

4. EXPERIMENTAL PHASE

Initially we will study the FTP technique in terms of Transfer Time, CPU Consumption and Memory Utilization. At first we will study Xen and then other Hypervisors.

4.1 FTP Technique

4.1.1 Transfer Time

Xen-PV is a hypervisor which uses Para-virtualization approach. It means that this hypervisor is located above the hardware. GuestOS called DomU shares the same resources by using Virtual Drivers [19], [20]. We want to transfer a ISO file = 600 MB from the client FTP virtual machine to the server FTP virtual machine. We have used SambaTool (which is part of Ubuntu 10.04 Server), to measure the Transfer Time for all the hypervisors.

Xen-HVM is a Full virtual machine. In order to use this hypervisor we should have a hardware that support it. As we saw in section 3 the parameters of the computers used in the experiments are optimal for the requirements [9]. Also is needed to build QEMU on Xen, so we should emulate the hardware in user space [10,11]. The Full virtualization in Xen has the same characteristics as VMWare which means that we can built OS with different native nature and different architecture, such as Windows in DomU. Also in Full virtualization technique it is not necessary to modify kernel OS Host or Guest. Nevertheless Full Virtualization has some disadvantages such as the increase of access time of I/O disks, because there are needed two trap instructions to access a disk [12]. The Full Virtualization includes an additive complex layer presented by QEMU emulation software. In order to emulate network drivers in both GuestOS we should install e1000 emulator in /root directory. The evaluation of Transfer Time in XEN-HVM is done using Samba FTP tool.

OpenVZ is a Hypervisor which operates in Operating System Level. Each GuestOS shares the same kernel with HostOS. To evaluate the Transfer Time of .iso file = 600 Mb between two virtual machines in different hosts we have used Samba FTP Tool.

KVM-FV is a Full virtualization approach. It is implemented by modifying Linux Kernel module. In our tests each virtual machine network driver has emulated from e1000 driver.

KVM-PV means that we should modify the kernel of HostOS. So we should install the *virtio driver* for each GuestOS. KVM-PV may improve the performance of KVM especially in terms of Transfer Time. For both cases we have used Samba FTP Tool.

4.1.2 CPU Consumption

Xen-PV. To evaluate the CPU consumption we used xentop –b command. But during the test we got just the flash result only for a moment. In order to get the average result for some CPU consumption values we created a script in C called ***AverageXen*** which records the values of xentop command every 5 seconds, then it presents the average value by using *avge* function. This script is located at /etc directory.

Xen-HVM. To evaluate the CPU consumption we used the same command and script as Xen-PV.

OpenVZ. To evaluate the CPU consumption in OpenVZ of FTP server we don't have any specific tool nevertheless we can measure the CPU wasted time in /proc/vz/vstat. To evaluate the CPU consumption we have created a script in C which is called ***traceproc1*.** It traces the active and idle processes in hypervisor by scanning the status of each process in vstat file. Each process has a wake bit in Process Status Register, if it is 1 this process is active and if it is 0 the process is idle. In ***Traceproc1*** script, located in /proc/vz, we have implemented a formula:

The availability of the process= (Time for each active process)/(Total CPU time) x100% (1)

The sum of the availability active processes = CPU Availability (2)

In reality this formula doesn't calculate the CPU availability, because when the processes are idle they still spend CPU time, consequently their consume CPU. Thus for the idle process we should build a semaphore variable [13] in order to make them sleep. In this way they will not consume CPU. Semaphore variables are built in a script in C called ***semaphore***, which records the ID of all idle processes. This information is taken from ***Traceproc1*** script. For each passive process we generate a thread which sends a signal to these processes. In this order, the passive processes are transformed in sleep processes. At the moment when CPU sends an interrupt message for one of the sleeping processes, the semaphore script is the first that takes this signal. This script reads the ID of calling processes, records it in a specific address into a specific register and then calls the specific thread. The thread wakes up the sleeping process. Thus the process can take the interrupt launched from CPU. This is a very dangerous approach because the script is implemented in user space, it means that after the interrupt request from CPU, the generated thread can't wake the process up. So the process is going to sleep forever. Nevertheless after this modification to ***traceproc1*** script we will evaluate the CPU consumption by using the formula:

CPU consumption= Sum of active processes/ Total nr of process (3)

We should emphasize that this script gives us an approximate value of CPU consumption in OpenVZ hypervisor.

Traceproc1 starts the calculation at the moment we start the transfer. So it is connected with Samba FTP Tool. At the moment we give the command to execute this tool, we execute another script which called ***init*** and is located at /proc. This script is executed at the same moment when samba tool executes the process calling *samba* function.

So this script is used as a bridge between Samba FTP tool and ***tranceproc1***.

KVM-FV. To evaluate the CPU consumption for KVM hypervisor we have used a tool called SAR Utility. As xentop command in Xen Hypervisor, SAR utility offers just a value in a time. In order to evaluate the average CPU consumption during the transfer of 600 MB .iso file we create a script in C language located in /dev/kvm called ***AverageKVM*** which uses a function called *avge*.

KVM-PV. We followed the same conditions as in KVM-FV in order to evaluate the CPU consumption.

4.1.3 Memory Utilization

Xen-PV. We want to evaluate the physical memory utilization of FTP Server during the transfer of .iso image between two virtual machines in different physical hosts. Initially we have used Mem_Access [14], but now we have used this tool to evaluate the Mem-Utilization during file transfer. We have implemented this tool to a script which is written in C and is called MemC. For each 10 MB transfer from Server Machine to Client Machine it calculates the Memory Utilization by activating this tool. The result is saved in a record. This record is part of "My SQL" Data Base which is installed in the Server Machine (mysql Ver 12.21 Distrib 4.0.14, for pc-linux).

Average Memory Utilization = Total Sum of record / nr of records. (4)

Always we use Samba FTP server to transfer data from one machine to another one.

Xen-FV. We have used the same method we used with Xen-PV to evaluate the Memory Utilization.

OpenVZ. We have used a tool named **stream_tool** [8] to evaluate the memory utilization in FTP Server. This tool gives just a value during the transfer. In order to take an average value of memory utilization we created a Script called ***AverageOpen*** which is localized in /proc.

KVM-FV. To evaluate the memory utilization we have used an open source **Stress Tool** [3].

KVM-PV. We have used the same way as KVM-FV to evaluate the memory utilization.

4.1.4 Real-Environment.

Finally we evaluated the three above parameters in real environment. To evaluate the Total Transfer Time we used the same tool called **Samba FTP Tool**. To evaluate the CPU consumption of FTP Server we installed **sysstat** tool: *yum install systat* and than executed **mpstat**. In order to get more details about the average CPU consumption we installed **top tool** [15]. In order to evaluate only the CPU Consumption during the transfer we can generate a script located in /etc called ***AverageCPU*** which executes **top tool** at the moment that **Samba FTP Tool** starts. So a function of this script called *samba1* is connected with the execution of **Samba Tool**. **Samba Tool** calls ***AverageCPU*** script which immediately calls **top tool**.

To evaluate the memory utilization we can use *GNOME System Monitor*. We can check the performance of Memory utilization at the moment we start the execution of **Samba FTP Tool**. In order to evaluate the proper time when **Samba Tool** starts the execution, we can generate another script in C which is called ***StartTime*** and records the computer time of the last **Samba Tool** Execution.

We have presented all the results of the experiments in table 2. In this table Host 2 is the FTP Server.

Table 2. The Evaluation of the Performance for 6 types of Environments based on FTP Client/Server approach.

The Hypervisor	Transfer Time	CPU Consumption	Memory Utilization (Host 2)
Real Environment	32 sec	8%	MAX=711 MB of RAM (≈17 % of RAM)
XEN-HVM	66 sec	24%	MAX= 1,08 GB of RAM (≈ 26 % of RAM)
XEN-PV	55 sec	13%	MAX= 887 GB of RAM (≈ 22 % of RAM)
Open VZ	53 sec	17%	MAX= 1,08 GB of RAM (≈ 27 % of RAM)
KVM-FV	71 sec	25%	MAX= 1,48 GB of RAM (≈ 37 % of RAM)
KVM-PV	64 sec	21%	MAX= 1,36 GB of RAM (≈ 34 % of RAM)

4.2 HTTP Technique

4.2.1 Transfer Time

XEN-PV. We want to evaluate the transfer time from web server machine to client machine by using Httperf benchmark, ver. 0.9.0 [14],[16]. The basic operation of this benchmark is to generate a fixed number of HTTP requests and to measure the number of responses that come back from the server and the rate they arrived. Initially we have installed LAMP (Appache 2 and My SQL client , My SQL Server) in Web Server Virtual Machine (Ubuntu 10.04 Server). We want to evaluate the time duration after 60 requests from client machine to Web Server machine. Each request is equal to 10 MB file called index1.html and it is located in /etc/apache2. We choose 60 request in order to transfer the same size (600 MB) which is the value of the FTP File transferred.

XEN-HVM. From the client machine we ran: Httperf –server 192.168.1.8 –uri /index1.html –num-conn 60 –rate=10 –timeout 5.

This is a simple example of generating workloads into a web server machine 192.168.1.8 (H1- Virtual Machine) with 60 connections with 10 connections for second.

OpenVZ. We could use different tools for OpenVZ such as Siegen tool [17] but we thought it was more convenient to use httperf tool.

KVM-FV. We have used the same command and the same tool to evaluate the transfer time between 2 VMs on different hosts by using httperf tool above KVM Hypervisor.

KVM-PV. To evaluate the transfer time of some web requests we used the same condition as we did in KVM-FV.

4.2.2 CPU Consumption

Xen-PV. In order to generate different values of httperf we used a tool called autobench [18]. So we could trace the curve of CPU Consumption from 6 request to 60 request for 10 times. Another way to evaluate the CPU consumption was the use of xentop –b command and a script called ***XenProc*** which calculates the average values recorded (by a function called *avge*) from *xentop* command located to /proc directory, written in C language. This command starts at the moment that *httperf* benchmark takes the command to generate of requests in Web Server.

Xen-FV. To evaluate the CPU Consumption in Web Server Virtual Machine above Xen-HVM we followed the same way as for Xen-PV.

OpenVZ. To evaluate the CPU consumption we used httperf benchmark associated with autobench tool:

autobench --single_host --host1 192.168.9.1 --low_rate 6 -- high_rate 60 --rate_step 10 --timeout 5 --file index1.html. It means that at the first time httperf benchmark initiates 6 req/sec and continuing up to 60 req/sec. Each connection contains 10 requests. Each request is 10 MB.

KVM-FV. To evaluate the CPU consumption we used again httperf benchmark as we did in OpenVZ approach.

KVM-PV. The same method as KVM-FV

4.2.3 Memory Utilization

XEN-PV. To evaluate the memory utilization we have to know two parameters; the transfer time and how many MB should be transferred for a time (heap size). Referring to the above experiment ***Httperf –server 192.168.1.8 –uri /index1.html –num-conn 60 –rate=10 –timeout 5***, the total number of requests is 60 and each request transmit 10 Mb file with a pause = 5 sec. We have built another script in C called ***MemC*** which gets information from **MemAccess** [14] benchmark for every request.

We have presented the steps of MemC algorithm:

a. *10 MB transmitted from Web Server to Client Machine by user command (httperf)*
b. *MemAccess tool records the memory utilization for one time*
c. *This result is saved in a file*
d. *This is a source file in MemC script*
e. *Find the average of the results*
f. *Send a periodic command (every 5 sec.) to MemAccess tool in order to record the new result in the source file*
g. *New record has came in source file.*
h. *Average results in Destination file*
i. *Loop for each 5 sec.*
j. *Final results are print in display*

This script calculates the results taken in Apache 2 installed in our machine. This script is implemented in order to include the **MemAccess** tool for each request. The results present the Memory utilization in Web Server Virtual Machine for each process which is located in the host computer. The total results can be presented in percentage. Each request from the Client Machine to the Server Machine are performed by using Httperf Benchmark which can generate 10 request in Second.

XEN-FV. We can evaluate the Memory Utilization in the same order as we did with XEN-PV

OpenVZ. We can evaluate the Memory Utilization as we did in Xen with the same tool called **MemAccess** but ***MemC*** script now is located in /proc/vz.

KVM-FV and **KVM-PV**. Script ***MemC*** is built over the /dev/kvm. We have used the same benchmark called **MemAccess** and the same tool in order to evaluate the Memory Utilization in Web Server called httperf.

4.2.4 Real Environment.

To evaluate the Transfer Time , CPU consumption and Memory Utilization in Apache Web Server which is installed on host computer with no hypervisor while the requests are transmitted between physical hosts we can use some commands such as: *cat /proc/cpuinfo* which gives the information about the number of processor and their status or *vmstat 5 10* which runs vmstat every 5 seconds for 10 iterations. This offers the number of process running and process block. In CPU section it offers the percentage of CPU consumption. By using this

command we get the memory utilization such as (memory swap, free memory, buffer cache etc). To evaluate the total transfer time, memory utilization and CPU consumption we have used SPECweb96. This benchmark is a workload generator and file set generator. It simulates Web Clients by sending Http request commands to the Web Server. The workload generator can measure the total transfer time (also theresponse time, but we do not analyse this feature). As a file set generator it can generate files with different sizes. Nevertheless we want to generate files (request) with 10 MB from Server Machine to Client Machine. So we measured the CPU consumption and Memory Utilization of Web Server. We incorporated AIX tool in SPECweb96 in order to trace the activity of Apache. By using *rmss tool* [21] = 4 GB (size of RAM) we evaluated the CPU consumption. In order to evaluate the Memory Utilization we used MemAccess tool [14], [21] in SPECweb96 benchmark.

Table 3. The Evaluation of Performance for 6 types of Environments based on HTTP Client/Server approach.

The Hypervisor	**Transfer Time**	**CPU Consumption**	**Memory Utilization (Host 2)**
Real Environment	26 sec	6 %	MAX= 589 MB of RAM ≈ 15 %
XEN-HVM	58 sec	22 %	MAX= 1,25 GB of RAM ≈ 31 %
XEN-PV	51 sec	12 %	Host 2 MAX= 1,12 GB of RAM ≈ 28 %
OpenVZ	50 sec	16 %	MAX= 1,05 GB of RAM ≈ 26 %
KVM-FV	66 sec	23 %	MAX= 1,32 GB of RAM ≈ 33%
KVM-PV	58 sec	20 %	MAX= 1,2 GB of RAM ≈ 30 %

5. CONCLUSIONS AND FUTURE WORK

1. Regarding to Transfer Time, the real environment has a performance approximately 2 times better than other hypervisors. It means that the Hypervisor includes a consider time delay during FTP transmission.

2. The real environment has a better performance in CPU consumption and Memory Utilization than the Hypervisors.

3. The Time Transfer in OpenVZ_FTP Server is smaller than that of the other hypervisors. This happens because OpenVZ hypervisor is less complex than the others. In OpenVZ each GuestOS has the same kernel as HostOS and each Guest looks like a simple process.

4. XEN-PV_FTP Server has a better performance in CPU consumption (13), because XEN-PV has a better isolation between GuestOS and HostOS.

5. The Memory Utilization in XEN-PV_FTP Server is better than the memory utilization in other Hypervisors. The second one is OpenVZ Hypervisor. The worse Memory Utilization is KVM_FTP/Web Server. This is because KVM offers more overhead than other Hypervisors [3], [4].

6.The Transfer Time in OpenVZ and XEN-PV Web Server is better than that of other hypervisors.

8. CPU consumption in KVM–PV is better than in KVM-FV because of the implementation of "virtio driver" in KVM-PV.

In the future we want to test the isolation and migration of these environments between virtual machines in different host in LAN and WAN.

6. REFERENCES:

[1] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hanseny, Eric July, Christian Limpach, Ian Pratt, Andrew War_eld, "Live Migration of Virtual Machines",2010

[2]. http://en.wikipedia.org/wiki/Live_migration

[3] Daniel Schlosser, Michael Duelli, and Sebastian Goll, "Performance Comparison of Hardware Virtualization Platforms" , 2010

[4] Nathan Regola, Jean-Christophe Ducom, "Recommendations for Virtualization Technologies in High Performance Computing", 2010

[5] Jianhua Che, Qinming He, "Performance Combinative Evaluation of Typical Virtual Machine Monitor", 2010

[6] Todd Deshane, Zachary Shepherd, Jeana N. Mathews," Quantitative Comparison of Xen and KVM", 2010

[7] Andrew Tanenbaum, "Modern Operating System" 4-th edition, chap 2 processes and threads, 2009

[8] Lucas Nussbaum, Fabienne Anhalt, Olivier Mornard, Jean-Patrick Gelas "Linux-based virtualization for HPC clusters", 2009

[9] Nathan Regola, Jean-Christophe Ducom, "Recommendations for Virtualization Technologies in High Performance Computing", 2010

[10] Chunqiang Tang, "High-Performance Virtual Machine Image Format for Cloud", 2008

[11] Daniel P. Berrang´e,,," Taking full advantage of QEMU in the Xen userspace", 2007

[12] Andrew Tanenbaum," Modern Operating System" 4-th edition, chap 1, Virtual Machines, 2009

[13] Andrew Tanenbaum, "Modern Operating System" 4-th edition, chap 2 processes and threads, 2009

[14] Jin Heo, Xiaoyun Zhu, Pradeep Padala, Pradeep Padala "Memory Overbooking and Dynamic Control of Xen Virtual Machines in Consolidated Environments", 2009

[15] http://msdn.microsoft.com/en-us/library/ms859476.aspx

[16] D. Mosberger and T. Jin. "httperf: A Tool for Measuring Web Server Performance. *Performance Evaluation Review*", 26(3):31–37, Dec. 1998

[17] Espen Braastad, "Management of high availability services using virtualization", 2006

[18].http://www.scribd.com/doc/25285257/Server-Consolidation-using-OpenVZ-Performance-Evaluation

[19] http://www.dell.com/downloads/global/power/ps3q05-20050191-Abels.pdf

[20] Lamia Youseff, Rich Wolski, Brent Gorda, Chandra Krintz. "Paravirtualization For HPC Systems", 2009

[21] Yiming Hu, Ashwini Nanda, Qing Yang, "Measurement, Analysis and Performance Improvement of the Apache Web Server", 2009

[22] James Curose, "Network Computer" Chap 2, Fifth Edition, 2009